Lab 2 – Linked List

The following SingleLinkedList interface is applied to questions 1 to 4.

```
struct Node {
    public:
        int data;    // value of list element
        Node *next; // pointer to next element of the list
}

class SingleLinkedList {
public:
    Node *pHead; // pointer to the 1st node of the list

    SingleLinkedList () {
        pHead = NULL;
    }

    void prepend(int data) {
        Node *pNew = new Node();
        pNew->data = data;
        pNew->next = pHead;

        pHead = pNew;
        return;
    }

    void display() {
        // add your code here
    }

    void insert(int data, int idx) {
        // add your code here
    }

    Node *search(int target) {
        // add your code here
    }

    void remove(int target) {
        // add your code here
    }

    void extend(SingleLinkedList other) {
        // add your code here
    }
}
```

Question 1: Use the already implemented method *prepend* to construct linked list L1 as follow:

L1 = {1, 9, 6, 5, 7, 10, 13, 4, 8, 7}

Then, implement method display to check your results.

Answer:

```
int main() {
    cout << "Question 1:" << endl;
    SingleLinkedList L1;
    int q1[10] = { 7, 8, 4, 13, 10, 7, 5, 6, 9, 1 };
    for (int i = 0; i < 10; i++) {
        L1.prepend(q1[i]);
    }
    L1.display();
}
```

Question 2: Implement method *insert* to add a new node with value ‘data’ at a given index ‘idx’.

e.g.

```
// L2 = {1, 3, 2, 5, 6}
L2.insert(4, 2) // data = 4, idx = 2
// L2 = {1, 3, 4, 2, 5, 6}
```

Answer:

```
void insert(int data, int idx) {
    if (idx > count) {
        cout << "Index is out of range." << endl;
        return;
    }
    if (idx == 0) {
        prepend(data);
    } else {
        Node *pNew = new Node();
        pNew->data = data;

        Node *pTemp = pHead;
        for (int i = 0; i < idx - 1; i++) {
            pTemp = pTemp->next;
        }
        pNew->next = pTemp->next;
        pTemp->next = pNew;
    }
    count++;
    return;
}
```

Question 3: Implement method *search* to find a node with value ‘data’.

e.g.

```
// L3 = {1, 3, 2, 5, 6}
Node *target = L3.search(5)
// target->data is 5
```

Answer:

```
Node *search(int target) {
    Node *pTemp = pHead;
    while (pTemp != NULL && pTemp->data != target) {
        pTemp = pTemp->next;
    }
    return pTemp;
}
```

Question 4: Implement method *remove* to delete ALL nodes with value ‘data’.

e.g.

```
// L4 = {1, 3, 2, 5, 6}
L4.remove(3)
// L4 = {1, 2, 5, 6}
```

Answer:

```
void remove(int target) {
    Node *pTemp = pHead;
    Node *pPrev = NULL;
    while (pTemp != NULL) {
        if (pTemp->data == target) {
            pPrev->next = pTemp->next;
            delete pTemp;
            pTemp = pPrev->next;
        } else {
            pPrev = pTemp;
            pTemp = pTemp->next;
        }
    }
    return;
}
```

Question 5: Implement method *extend* to join two linked list.

e.g.

```
// L5a = {1, 4, 7}
// L5b = {9, 6, 5}
L5a.extend(L5b)
// L5a = {1, 4, 7, 9, 6, 5}
// L5b = {9, 6, 5}
```

Answer:

```
void extend(SingleLinkedList other) {
    count = count + other.count;
    if (pHead == NULL) {
        pHead = other.pHead;
        return;
    }

    Node *pTemp = pHead;
    while (pTemp->next != NULL) {
        pTemp = pTemp->next;
    }
    pTemp->next = other.pHead;
    return;
}
```

The following struct is used to form DoubleLinkedList

```
struct Node {
    public:
        int data;
        Node *next;
        Node *prev;
}
```

Question 6: Create a new class DoubleLinkedList and implement the corresponding method *insert* and *remove* same as stated for SingleLinkedList.

Answer:

```
class DoubleLinkedList {
private:
    TwoWayNode* pHead;
    int count;
public:
    DoubleLinkedList() {
        pHead = NULL;
        count = 0;
    }
    void prepend(int data) {
        TwoWayNode* pNew = new TwoWayNode();
        pNew->data = data;
        pNew->prev = NULL;
        pNew->next = pHead;
        pHead = pNew;
        return;
    }
    void insert(int data, int idx) {
        if (idx > count) {
            cout << "Index is out of range." << endl;
            return;
        }

        if (idx == 0) {
            prepend(data);
        } else {
            TwoWayNode* pNew = new TwoWayNode();
            pNew->data = data;
            TwoWayNode* pTemp = pHead;
            for (int i = 0; i < idx - 1; i++) {
                pTemp = pTemp->next;
            }
            pNew->next = pTemp->next;
            pNew->prev = pTemp;
            pTemp->next = pNew;
            if (idx == 0) {
                pHead = pTemp;
            }
        }
        count++;
        return;
    }

    void remove(int target) {
        TwoWayNode *pTemp = pHead;
        TwoWayNode *pDel = NULL;
        while (pTemp != NULL) {
            if (pTemp->data == target) {
                pTemp->prev->next = pTemp->next;
                pDel = pTemp;
                pTemp = pTemp->next;
                delete pDel;
            } else {
                pTemp = pTemp->next;
            }
        }
        return;
    }

    ~DoubleLinkedList() {
        TwoWayNode* pTemp = pHead;
        while (pTemp != NULL) {
            pTemp = pTemp->next;
            delete pHead;
            pHead = pTemp;
        }
    }
};
```

Question 7: Implement method *reverse* for DoubleLinkedList.

e.g.
```
// L7 = {1, 3, 2, 5, 6}
L7.reverse()
// L7 = {6, 5, 3, 2, 1}
```

<u>Answer:</u>

```
    void reverse() {
        TwoWayNode* pCurrent = pHead;
        TwoWayNode* pTemp = NULL;

        while (pCurrent != NULL) {
            pTemp = pCurrent->prev;
            pCurrent->prev = pCurrent->next;
            pCurrent->next = pTemp;
            pCurrent = pCurrent->prev;
        }

        if (pTemp != NULL) {
            pHead = pTemp->prev;
        }
        return;
    }
```